Student Research Poster: A Scalable General Purpose System for Large-Scale Graph Processing

Jiawen Sun*
Supervisors: Hans Vandierendonck and Dimitrios S. Nikolopoulos
Queen's University Belfast, University Road, Belfast,UK BT7 1LR
jsun03@qub.ac.uk

1 Introduction

Graph analytics is an important and computationally demanding class of data analytics. It is essential to balance scalability, ease-of-use and high performance in large scale graph analytics. As such, it is necessary to hide the complexity of parallelism, data distribution and memory locality behind an abstract interface [2].

Our aim is to build a scalable graph analytics framework that does not demand significant parallel programming experience based on NUMA-awareness. The realization of such a system faces two key problems: (i) how to develop a scale-free parallel programming framework that scales efficiently across NUMA domains; (ii) how to efficiently apply graph partitioning in order to create separate and largely independent work items that can be distributed among threads.

2 Graph Processing Model

We address the scalability issue through extending parallel programming models using numa-aware for parallelism. In the task dataflow model all data touched by a task is explicitly annotated, along with read/write side effects. This allows the runtime to schedule tasks during execution while respecting dataflow dependencies [4].

We build on Swan [4], a Cilk extension that supports dataflow with NUMA-awareness. This allows executing the Ligra graph analytics system [3] unmodified in shared memory. Ligra optimizes the graph traversal method depending on whether the frontier, the set of currently active vertices, is densely populated or sparsely populated. If densely populated, it scans over all vertices to locate active vertices in either a *backward* or a *forward* traversal [3]. If sparsely populated, it performs direct access to the active vertices in *forward* mode.

3 Graph Partitioning

We extend Ligra [3] to partition graphs using the same strategy as GraphChi [1] and Polymer [5]. This partitions the set of edges and *replicates* vertices across partitions.

Partitioning introduces a significant number of vertices with zero in- or out-degree. We extend the graph representation by adding to each vertex the index of the next vertex with non-zero degree. This allows us to skip ahead during graph traversal and improves performance by 11.5% on average. We furthermore optimize the calculation of the frontier. We adapt the traversal routines to operate on either sparse or dense frontiers to avoid conversion of the frontier.

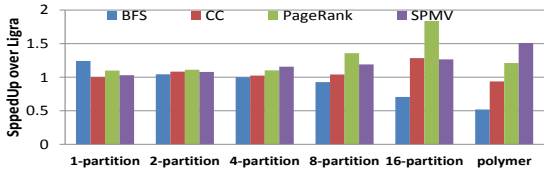

Figure 1: Speed-up over Ligra for twitter graph (96 threads). N-partition is our optimized partitioning method, No-NUMA-Optimization, polymer uses 4-partition, twitter graph.

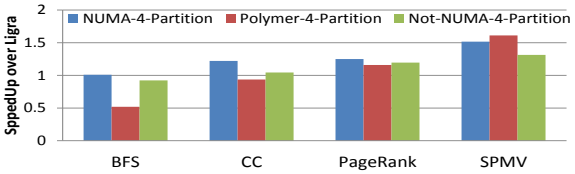

Figure 2: NUMA optimization of 4-partition across 4 NUMA node machine (96 threads), twitter graph

Preliminary results are on a quad-socket 2.6GHz Intel Xeon E7-4860 processor,totaling 96 cores. They show that partitioning increases instruction count for *dense backward*, which affects BFS and CC. While Polymer performs worse on CC compared to Ligra, our optimization improves performance over Ligra. The partitioning furthermore improves locality during the *dense forward*, which explains improved performance for PageRank and SPMV (Figure 1, 2). After NUMA-aware scheduling, we could get better performance.

In future work we will further investigate graph partitioning to improve performance and refine the scale-free programming model.

4 References

[1] A. Kyrola, G. E. Blelloch, and C. Guestrin. GraphChi: Large-scale graph computation on just a PC. In *OSDI*, 2012.

[2] G. Malewicz *et al.* Pregel: a system for large-scale graph processing. In *SIGMOD*, 2010.

[3] J. Shun and G. E. Blelloch. Ligra: a lightweight graph processing framework for shared memory. In *PPoPP*, 2013.

[4] H. Vandierendonck, G. Tzenakis, and D. S. Nikolopoulos. A unified scheduler for recursive and task dataflow parallelism. In *PACT*, 2011.

[5] K. Zhang, R. Chen, and H. Chen. NUMA-aware graph-structured analytics. In *PPoPP*, 2015.

PACT '16 September 11-15, 2016, Haifa, Israel

DOI: http://dx.doi.org/10.1145/2967938.2971465